Math

UIUCDCS-R-73-605

SIMULATION ANALYSIS OF A PAY-FOR-PRIORITY SCHEME FOR THE IBM 360/75

BY

Sandra Ann Mamrak

August 1973

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

UIUCDCS-R-73-605

SIMULATION ANALYSIS OF A PAY-FOR-PRIORITY SCHEME FOR THE IBM 360/75

BY

SANDRA ANN MAMRAK

August 1973

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

This work was supported in part by NSF GJ 28289 and was submitted in partial fulfillment of the requirements for the Master of Science degree in Computer Science at the University of Illinois at Urbana-Champaign.

ACKNOWLEDGMENT

The constant support and expert guidance of Professor E. K. Bowdon, Sr. were essential elements in the preparation of this paper. I would like to sincerely thank him.

I would like to acknowledge Mr. Fred Salz for his assistance in developing the benchmark jobstream and his assistance in modifying his basic GPSS simulator to suit the purposes of this project. Mr. Robert Skinner has also rendered invaluable aid during our many discussions relating to the design philosophy and the progress of the project. Mrs. Gayanne Carpenter's willing assistance and superb typing were the last, but surely not the least, contribution to the completion of this paper.

This research was supported in part by the Computing Services Office at the University of Illinois at Urbana-Champaign and in part by the National Science Foundation under Grant No. NSF GJ 28289.

PREFACE

When a computing facility becomes so heavily utilized that dissatisfaction about long turnaround times is prevalent in the user community, the introduction of a priority consideration into the scheduling algorithm will aid in lessening the problem. Users who desire shorter turnaround time may then opt for a high priority position on the job queue and users with less urgent jobs may opt for a middle or a low priority position on the job queue. High and low priority queue positions may be associated with higher and lower rates of job cost respectively.

A responsible introduction of a priority scheduling algorithm into a given system must include a serious consideration of the goals of such an algorithm as they apply to that system, a thorough study of the various priority scheduling options available to the system, and the developing and analyzing of models embodying the most favorable of these options.

The results of the development and evaluation of a priority scheduling algorithm for the University of Illinois computing center are presented in this paper. Project goals are set forth, algorithm options are considered and a thorough discussion of the formulation and analysis of various simulation models of proposed schemes is included. Recommendations are made concerning the follow-up procedures required to make the project successful.

TABLE OF CONTENTS

1. INTRODUCTION: RESEARCH DESIGN

At the University of Illinois at Urbana-Champaign, the present scheduling algorithm used in the IBM 360/75 run under HASP 3.1 and O.S. [1] queues jobs for service according to their resource requests. A brief description of this scheduling algorithm can be found in Appendix A. Very important tasks must assume queue positions determined by their particular set of resource requests, irrespective of their urgency. A priority algorithm is desired, therefore, that allows jobs to vie for queue position depending on how short a turnaround time the user desires for his job. The basic principle behind such an algorithm is that users would be able to choose a priority for their jobs and would be charged for service accordingly. If a user is willing to pay more than the normal rate his job would be given a higher priority; if a user wishes to pay less than the normal rate, his job would be given a lower priority. This sort of scheduling algorithm is referred to as "pay-for-priority".

As the first step in the development of a pay-for-priority scheme, specific objectives of such a scheme as it would be implemented by the Computing Services Offices for the University of Illinois user community were determined. These goals were intended to be a set of standards against which proposed pay-for-priority schemes could be tested and evaluated. In addition, a study was made of many possible options available in the design of a pay-for-priority algorithm.

Eight specific models incorporating various combinations of these options were selected to study. These eight models were built upon two basically dichotomous schemes--one in which discrete priority assignments were made and one in which continuous assignments were made. Relatively simple versions of each of the two schemes were cumulatively embellished with the addition of more finely tuned algorithm options.

A GPSS simulation of the IBM 360/75 was used [2] to implement and evaluate the specific pay-for-priority schemes. In order to use the GPSS simulator to predict real system performance within well-defined statistical limits of accuracy, the simulator had to be tuned with data characteristic of the environment to which the results were to be applied. It was necessary, therefore, to create a meaningful benchmark jobstream representing the workload of the period under study. Formulating a tuned simulator from the various system performance parameters produced by running the benchmark, and then validating the accuracy of the simulator were preliminary to any actual pay-for-priority algorithm evaluations.

As the final step in the pay-for-priority scheme development and evaluation, simulation runs of eight specific models were performed and analyzed, yielding data upon which a report of comparative algorithm performance could be based.

2. OBJECTIVES OF A PAY-FOR-PRIORITY SCHEME

The set of goals to be met by any specific pay-for-priority scheme will be as unique to an organization as are its service philosophy and its user community. A university computing facility in particular serves a wide range of users representing an equally wide range of system demands. The objectives of a pay-for-priority scheme as listed below are calculated to meet the diverse needs of users serviced by the Computing Services Office at the University of Illinois. These goals are standards by which any pay-for-priority scheme devised for the university community must be measured.

OBJECTIVE 1: User Control

> To allow users to influence their scheduling priority, and thus the expected turnaround time, for each job submitted to the system.

Involved in this objective is giving the user the ability to intelligently influence the turnaround time for his job by choosing an appropriate priority for it. Either a static or dynamically updated report can be issued informing the user of turnaround time for jobs in the various priority levels. The user can then be expected to determine priority parameters for his job that will place it, with its particular resource requests, in the desired priority level.

OBJECTIVE 2: System Efficiency

> To maintain as high a level as possible of resource utilization and system throughput.

Involved in this objective is the degree to which system resource utilization and throughput will be permitted to degrade as a result of disregarding the normal scheduling algorithm in favor of choosing jobs strictly according to their paid for priority. The goal is to design a scheme that will dynamically maintain some acceptable level of resource use and throughput and will compensate in monetary returns for any required deviations from these acceptable levels.

OBJECTIVE 3: Load Leveling

> To evenly distribute an essentially varying workload over the 24-hour work day and over longer periods displaying fluctuations in use such as weekly, monthly, and semester periods.

Involved in this objective is the need to provide incentives for users which would encourage them to use the system at times that presently are periods of low level usage. These low level usage times must be viewed in terms of hours of the day, i.e. midnight to 5 a.m., days of the week, weeks of the month, and months of the semester. Conversely, the reward system must be such as to discourage the running of less urgent jobs during periods of high level usage.

OBJECTIVE 4: Predictability

> To dynamically predict with reasonable accuracy the expected turnaround time for jobs with given priorities.

Involved in this objective is the requirement that the priority scheme should dynamically provide the user with reliable information about the turnaround time that can be expected for a job if that job is submitted at a particular priority level. This information must be regularly updated and reflect the current turnaround times for the different priority levels.

OBJECTIVE 5: Easy Use

To be easily understood by users.

Involved in this objective is the necessity of the adopted priority scheme to be readily understood by the average user and to be able to be used easily and intelligently, with little effort and adequate rewards.

OBJECTIVE 6: Practicality

To be practical to implement.

Involved in this objective is the question of ease and speed of implementation of a proposed priority scheme. The scheme should be able to be introduced into the system by system programmers within a reasonable amount of time and be maintained and adjusted in a straight-forward manner after its implementation.

OBJECTIVE 7: Fairness

To guarantee all jobs, regardless of priority, the opportunity to actively compete for O.S. resources during some allotted time period.

Involved in this objective is the right of users to be guaranteed some processing time on the computer, even if they choose a lower priority level than other users. No users are to be excluded from system use solely as a result of implementing the pay-for-priority scheme. Further, users may expect an acceptable turnaround time for a job submitted to the computing facility, independent of their ability to pay more than the normal service rate.

3. OPTIONS IN A PAY-FOR-PRIORITY SCHEME

Within the framework of the IBM 360/75 under HASP and O.S., there exist a myriad of options in formulating a pay-for-priority scheme. These options can be generally described in four basic sets of choices, each of which allows for several variations within itself.

An underlying assumption in the options described below is that all jobs entering the system are subject to the pay-for-priority scheme, participating with default values when the specific pay-for-priority parameters are not assigned values by the user.

OPTION 1: Continuous or Discrete Assignments

> A continuum of scheduling priority assignments and corresponding rates of pay-for-priority vs. a discrete class assignment of priority levels and corresponding rates of pay.

Present job class assignments and their resulting quasi-priority assignments are made based on some estimate of a job's size as reflected in the resources it requests. Involved in Option 1 is the choice of either implementing a continuum of priority assignments or of implementing a set of discrete levels of priority assignments within the framework of the present job class scheme.

In the former case, an initial priority assignment based on a job's resource requests would be given to each job and the pay-for-priority assignment would be made using the job determined priority and a user determined reward factor r. For example, the pay-for-priority assignment might be given by the quotient of the initial priority and the reward factor.

When processing is completed, the charges for each job can be computed by multiplying the normal system charge units times the factor r (or some function of r). This scheme allows a job's priority assignment to be dependent on its size as well as on the rate the user is willing to pay for its priority.

The latter choice in this option is to maintain the basic class structure as it is presently used and to manipulate a job's priority either by assigning it to a special class P which would have dedicated initiators, or by assigning it to classes A-D and grading its priority within these classes--i.e. 1.1, 1.2, . . ., 13.9, 14.0--according to the rate the user is willing to pay.

OPTION 2: Dedicated or Competitive Resource Use

> Assignment of various percentages of dedicated O.S. resources to jobs at different priority levels vs. job competition for O.S. resources as is presently implemented on the system.

Once a job is in O.S., its processing time is affected not only by its own resource needs, but also by the other jobs that are simulataneously making demands on O.S. resources. The choice involved in Option 2 is one of either implementing a scheme whereby percentages of O.S. resources (CPU, I/O channels, core) would be dedicated to jobs of certain priority levels [3] or O.S. resources would be assigned as is presently done.

In the former case a reference table can be established indicating what percentage of O.S. resources would be dedicated. For example, one such table might contain the following:

Table 1. Percentages of Dedicated OS Resources

Level of Priority	% CPU	% Core	% I/O
> 100% normal rate	50	60	50
= 100% normal rate	35	30	25
< 100% normal rate	15	10	25

The scheduler can reference a dynamically kept record of present system resource usage and choose its next job so as to maintain as closely as possible the levels of usage defined in the table.

For the second choice in this option, even though jobs might gain or lose position on the HASP queue as a result of their particular priority assignment, once jobs are in O.S. they would vie for O.S. resources in exactly the same way as they do now.

OPTION 3: Workload Dependent or Static Assignments

> Dynamically adjust pay-for-priority rates and percentages of dedicated O.S. resources with an increasing workload vs. keep static priority rate and dedication assignments.

Accepting large jobs for processing during peak system loads may cause a degradation in resource utilization and system throughput. Involved in Option 3 is the choice to either increase the cost for faster service, along with adjusting percentages of dedicated O.S. resources or to keep static pay-for-priority and dedication percentages, regardless of the system load.

In the former case, a job's priority can be computed using the job determined priority and user determined reward factor as described earlier. The cost of the job, however, instead of being just a function of resources used and the reward factor r, would also include a multiplication factor which would measure the system workload. The workload factor can be adjusted such that jobs requesting no special priority considerations would pay the normal rate, jobs requesting a high priority would pay increasingly higher rates (depending on the workload) and jobs requesting a low priority would pay increasingly lower rates.

In addition to adjusting job costs for varying workloads, O.S. resource allocation can also be adjusted so that dedication of CPU, Core, and I/O facilities would vary with the workload. In this case, Table 1 might be transformed into the following:

Table 2. Dynamic Assignment of Percentages of Dedicated OS Resources

Level of Priority	Low Level Workload % CPU/Core/IO	Medium Level Workload % CPU/Core/IO	High Level Workload % CPU/Core/IO
> 100% normal rate	45/50/45	50/60/50	60/65/65
= 100% normal rate	30/30/30	35/30/25	40/35/35
< 100% normal rate	25/20/25	15/10/25	

For the second choice in this option, pay-for-priority rates and/or dedication percentages would not change in a dynamic way, but would stay set until some external intervention altered them.

OPTION 4: Ageing or Absolute Priority

> Increase a job's priority as a result of its waiting time in the system vs. maintain static priority assignments.

A job gains or loses position in the HASP queue as a result of other jobs being run or other jobs of higher priority joining the queue. Involved in Option 4 is the choice to either let a job also gain position in the HASP queue as a result of the time it has spent on the queue, thus allowing initial priorities to be overridden, or to let it move forward only as a result of jobs ahead of it being run.

4. DEVELOPMENT OF SPECIFIC PAY-FOR-PRIORITY MODELS

The four basic options for implementing a pay-for-priority scheme and the many variations possible within each option present the priority algorithm designer with a large number of schemes to be considered. A systematic approach organized around the basic dichotomy of the set of discrete algorithms and the set of continuous algorithms was adopted to simplify the study.

Each of two basic models (discrete or continuous priority assignments) was run with no pay-for-priority scheme options added so that a standard set of system performance parameters could be collected. These relatively simple models were then each embellished with the cumulative addition of pay-for-priority options. System performance parameters were collected after the addition of each option thus making comparisons of the different schemes possible. The design of the pay-for-priority algorithm evaluation is summarized in Table 3 and described in detail below.

4.1. Two Basic Schemes

The two basic models used to evaluate the pay-for-priority algorithms were both GPSS simulations of the IBM 360/75. The model used to represent the discrete priority assignment scheme simulated the operation of the IBM 360 under HASP and O.S. using the Magic Number and class assignment schemes as are used in the current system. The express initiator was inactivated and the remaining six initiators were set to

Table 3. Summary of Pay-for-Priority Models

Options Implemented	Discrete Algorithms	Continuous Algorithms
Simple priority scheme with turnaround prediction capabilities.	A basic simulator of the IBM 360/75 under HASP and O.S. was developed and a turnaround time prediction module was added to it.	A simulator of the IBM 360/75 was developed to run under a PRT priority scheme which assigns continuous priorities to jobs. The prediction module was added
Addition of pay-for-priority.	The Magic Number scheme is used with options for either a low, normal, or high priority within a given class.	The PRT scheme is used with options of low, normal, or high priorities.
Addition of an ageing option.	A module is added which bumps a job's priority when the job has been waiting in the system for a given period of time.	A module is added which increases a job's priority as a result of its being in the system a long time.
Addition of varying workload considerations.	A module is added which calculates a job's cost using a workload factor. High priorities cost more under heavy loads.	The same workload varying cost scheme module is added to the PRT simulator.

A, AB, BA, BA, BAC and CBA. The model used to simulate the continuous priority assignment scheme used a "PRT" assignment [4]. This scheme assigns a unique priority to each job based on the job's resource requests. The actual priority assignment, PRT, is defined by

$$PRT = \alpha \cdot IOREQ + CPUSEC + \beta \cdot REGION$$

where α and β are experimentally determined coefficients which measure the amount of time a job will spend in O.S. as a result of its I/O requests and core requests respectively. Jobs are queued in HASP in order of increasing PRT and all initiators are set to select as their next job the one with the lowest PRT.

4.2. More Complex Schemes

To these two basic models was added a turnaround time prediction module. In the Magic Number scheme the predicted turnaround time for a job in a given class is determined by averaging the actual turnaround times of jobs run in that class. This predictor is recalculated at two minute intervals, with the "seed" for the present two minute average being taken as the average turnaround time during the last two minute interval.

In the PRT scheme, as a job is terminated, its priority assignment and its turnaround time are saved in an array. The array is large enough to contain these points of information for the fifty most recent jobs completed. At two minute intervals a least squares curve-fitting

routine is used to calculate the coefficients of the first degree polynomial which best approximates the pairs of coordinates presently in the array. These coefficients are then used to approximate the expected turnaround time for all jobs arriving during the next two minute interval. Figure 1 illustrates the prediction module function, given nine pairs of hypothetical coordinates.

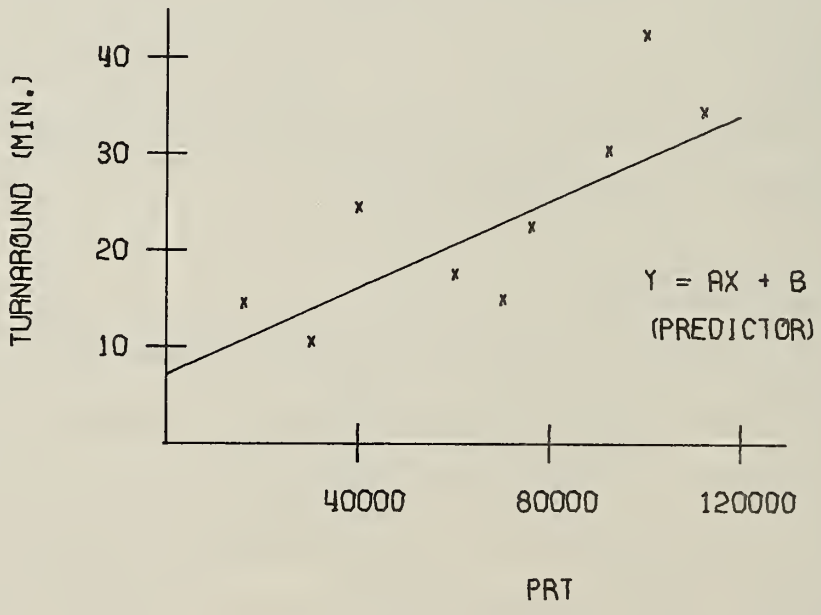


Figure 1. PRT Predictor

Pay-for-Priority Module

The pay-for-priority schemes under consideration begin with the simplest possible models and become more complex with the cumulative addition of options. Initially, a three level pay-for-priority scheme was added to both the Magic Number and PRT schemes.

In the Magic Number model, the user is permitted to choose either a high, normal, or low priority within his job's class. For the high priority, he is charged 150 percent the normal rate and for the low priority, he is charged 75 percent of the normal rate.

In the PRT scheme, the user is also permitted to choose a high, normal, or low priority for his job. The high priority is assigned by dividing the job's PRT by 6/5. The low priority is assigned by dividing the job's PRT by 5/6. Charges for the high and low priorities are the same as for the Magic Number scheme.

Ageing Option

Since a job that is paying for a high priority still has to contend for resources with other high priority jobs, and since low priority jobs should be serviced within some reasonable time after they enter the system, an ageing module was added to each of the two models to ensure some maximum upper limit on the time spent in the system. In the Magic Number scheme, a job is "bumped" to the next higher priority level depending on its waiting time as indicated in Table 4. Note that B-high is not bumped into A-low and C-high is not bumped into B-low.

Table 4. Ageing Parameters

A priority X job, after Y minutes spent in the system, will become a priority Z job.

X	Y	Z
A	30	A-high
A-low	45	A
B	75	B-high
B-low	90	B
C	120	C-high
C-low	145	C

In the PRT scheme, basically the same ageing algorithm was used, with the continuous PRT values being transformed into eight discrete uniform intervals. The "bumping" was accomplished by adjusting the job's priority such that it would fall into the next higher interval after an appropriate amount of waiting time.

Workload Considerations

Previous work done on a pay-for-priority algorithm [5] indicates that allowing large jobs requesting high priorities to disturb the normal scheduling algorithm during heavy workloads can result in a significant loss of system revenue and throughput. As a result of this observation, a module was added to the magic number and PRT schemes which calculates a job's cost based both on the priority under which it is run and also on a workload factor, h, which is indicative of the busyness of the system. In the Magic Number scheme, the workload factor is calculated

by considering the average turnaround time of the most recently completed jobs. If the turnaround time is high, the system is considered to be busy and if it is low, the system is considered to be light. In the PRT scheme, the workload factor is taken to be the slope of the predictor line. The steeper the slope, the more busy the system. Under this scheme, a job requesting high priority during a busy period pays more for the same work than it would when the system load was light. Conversely, a job requesting low priority pays less than it would when the system load was light.

Specifically, the Magic Number and PRT workload factors were implemented in the following way. In the Magic Number scheme, the workload factor h was determined as a job entered the system and was stored in a parameter associated with that job. The factor h was calculated by using a formula which divided the average number of minutes of waiting time for a respective class by either 1, 2, or 3, depending on whether the respective class was A, B, or C. For example, if the average waiting time for a class B job was currently 60 minutes, then a class B job entering the system would have an h factor associated with it equal to $\frac{60}{2} = 30$. If the average waiting time for a class A job was 60 minutes, then an entering class A job would have an h factor of 60 assigned to it. This h factor was used in calculating the cost for either a low or high priority request. For high priority requests, the h factor was added to 110 to form the percentage of the regular cost that a job would be charged. In the case of the class B job mentioned above, that job would be charged 140 percent of its normal cost. If the class B job requested a low priority

instead, the h value would be subtracted from 90 and, hence, in the class B example the charge would be 60 percent of the regular job cost. Limits were set on the h factor so that a high priority job never was charged more than 300 percent of normal cost and a low priority job was never charged less than 30 percent of normal cost.

The PRT workload factor was determined from the slope of the predictor line. The slope values were normalized to correspond approximately to the average turnaround time values used in the Magic Number scheme. The identical charging scheme was used.

The workload factor provides a negative incentive to users in that it tends to discourage high priority requests during periods of heavy system use. The workload factor feature also has the effect of disabling the pay-for-priority scheme during periods of light system use, since charges for low and high priority jobs would be more nearly equal to those for a middle priority job.

Other Options

Beside the various option combinations that were chosen for evaluation, there exist several other possibilities that could be considered. Chief among these is the option to use dedicated O.S. resources as a part of the pay-for-priority scheme. The evaluation of this option was omitted from this study because of the complexity required to simulate such an algorithm and the impracticality of actually implementing such an algorithm on the IBM 360/75 at the present time. Also, the PRT scheme could be run with a continuous spectrum of priorities available to the

user. In this case, the user specifies a reward factor r which determines both the job's priority and the cost. If r=1, the priority is the PRT and the cost is the normal rate. For r>1 or r<1, the new priority, PFP, is given by

$$PFP = PRT/r^2$$

and the cost is the normal rate multiplied by r.

5. EVALUATION OF THE SPECIFIC SCHEMES

Before the GPSS simulator could be used to test the proposed pay-for-priority models, it was necessary to verify that the simulator did indeed accurately reflect real system performance.

5.1. Validation of the Simulator

The simulator is made to model a particular operating environment of the IBM 360/75 by inputting the various distributions of system performance parameters (I/O, core, and CPU requests per job step, percent of class A, B, C, D, and X jobs, initiator settings, etc.) characteristic of that environment. The model is validated by demonstrating that the simulator will not only reproduce the input parameter distributions, but will also accurately predict other performance parameters characteristic of the same jobstream.

The first task in the validation process, therefore, was to create a jobstream which characterized the system environment in which the pay-for-priority results were to be applied.

The Benchmark Jobstream

Since scheduling algorithms based on priority assignments are, in fact, designed to be most effective when the system is busy, a benchmark tape was developed to be used to evaluate system performance of the IBM 360/75 during typical periods of heaviest use. A sampling procedure was defined with three basic sampling principles in mind--use as short a

sampling period as possible, minimize the clustering of the sampling periods, and sample over as many periods as possible--subject to the physical and practical limitations of time and cost considerations.

System records from the previous year containing the distribution of jobs run versus time of day were used to generate random times of the day at which samples were extracted. Given a total of four hours of real time over which to spread the sampling, two groups of 24 five-minute sampling periods during the hours of 10 A.M. to 6 P.M. on Mondays through Fridays were chosen. The first group of samples was collected from March 14 to March 18, 1973. The second group was collected from April 23 to April 27, 1973. Of all the possible time slots available in each of these testing periods, those time slots which showed the highest use in the previous year were given the greatest probability of being chosen.

Adjustments had to be made to the jobstream in order to run the sample set of jobs without re-using private files that had been read and/or written on by users whose jobs appeared on the benchmark sample. These adjustments were made by searching the system records for the statistics relating to actual runs of the jobs in question and determining exactly how many I/O requests and CPU centiseconds were used by each. The jobs were then replaced on the benchmark by "dummy" jobs which had their identical I/O and CPU characteristics. The substitute jobs represented about one third of all the usable jobs on the benchmark tape. Beside the substitutions, several jobs had to be deleted. Some jobs had

read and/or written on private files and then punched cards or used the plotter. It was not possible to run these jobs again. A total of 625 jobs out of an original 718 jobs were on the benchmark tape in its final form. (Express jobs were not collected or considered in this study.)

Real System Parameters

For the actual four hour run of the benchmark tape from which data was to be gathered for tuning and validating the simulator, the system was first seeded with fifteen jobs. Jobs were then read in and vied for resources at the rate of 100 per hour. System statistics were collected on the 414 jobs which completed in the four hour run.

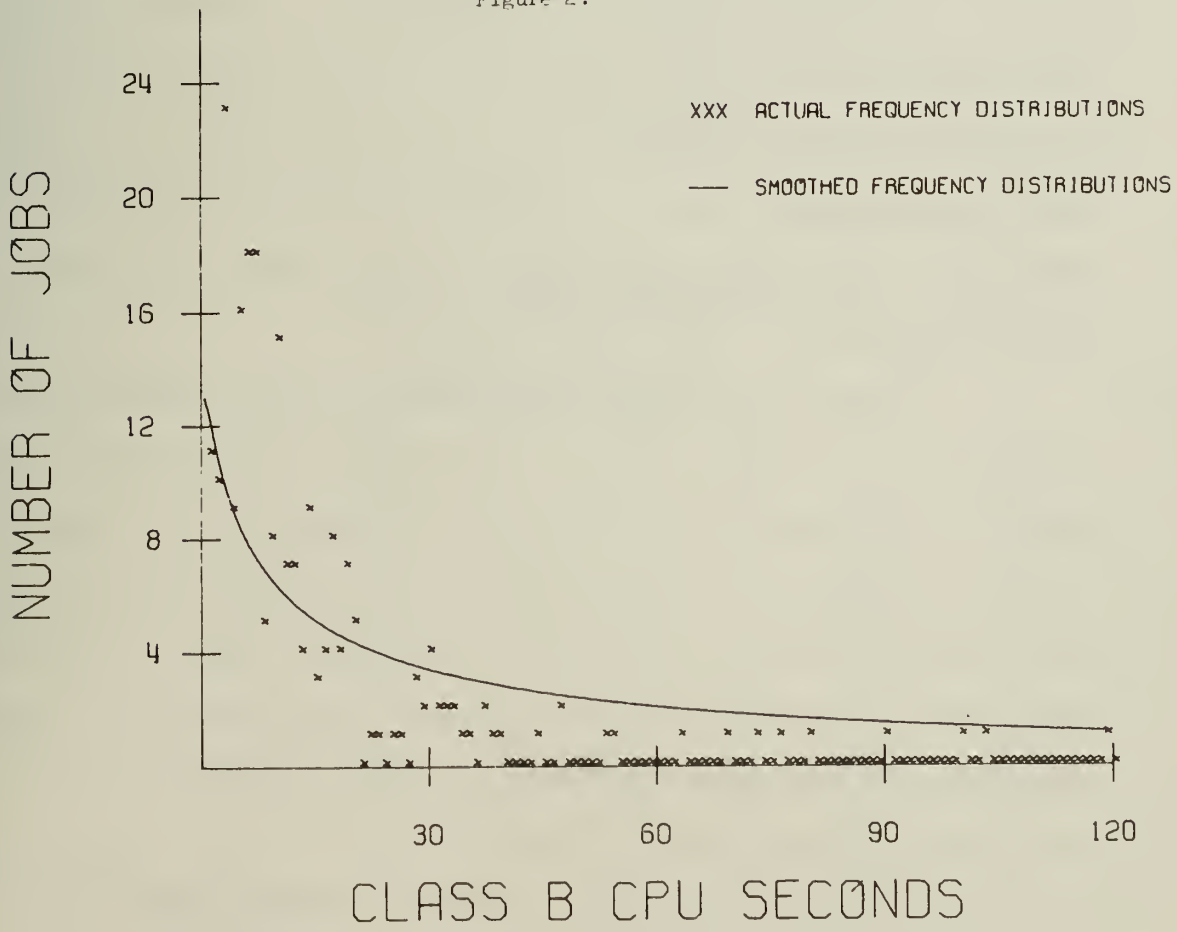
The parameters which were collected from the benchmark run for the GPSS simulation of the 360/75 included:

- frequency distributions of I/O requests and of CPU centiseconds by job class
- distributions of core requests by job step
- distributions of lines printed and of cards punched
- percentage of jobs having 1, 2, or 3 steps
- percentage of the total job I/O and CPU resources used by each job step
- percentage of jobs in the various classes A, B, and C

Frequency distributions were collected using a basic "bin" techinque. For each of the quantities over which a distribution was to be determined, 120 uniform bins were created and their size was determined by the spread of input for that particular quantity. The bins for class A CPU, for example, each represented 150 centiseconds of CPU time and ranged from 1-150 centiseconds to 17851-18000 centiseconds.

The core bins, on the other hand, each represented 3 kilobytes of core and ranged from 1-3 kilobytes to 358-360 kilobytes. Into each of the bins was accumulated the number of jobs on the benchmark which used the particular range of resources represented by the bin. A frequency graph of Class B CPU seconds is shown in Figure 2. Over the actual bin values is superimposed the smooth fitting curve that was determined using a least-squares fit technique.

Figure 2.



Because smoothed distributions represent a total population more accurately, the outputs of the various frequency distributions were fit to a smooth curve over the range of the respective data. An inspection of the frequency graphs of I/O and CPU for classes A-C, lines printed and cost indicated that the curves represented log-normal distributions. These curves are of the form:

$$f(x,\alpha,\beta) = \frac{1}{x\beta\sqrt{2\pi}} e^{\frac{-(\ln x - \alpha)^2}{2\alpha^2}} \qquad t>0$$

$$= 0 \qquad t\leq 0$$

Since the log-normal curves are so close to the exponential family, exponential curve-fits were also tried, but with poorer results than the log-normal fits.

An inspection of distribution graphs of core requests by job step indicated that these functions consistently hovered around common default values such as 58K or 116K, with few exceptions. The step core functions, therefore, were not smoothed with the curve-fitting package, but instead were fed into the simulator discretely, using the distribution values from the respective frequency tables. The distribution of cards punched per job was handled similarly since the distribution graph of this function was also highly irregular.

The curve fitting was done using a package program authored by J. A. Middleton titled, "Least-Squares Estimation of Non-Linear Parameters--NLIN" [6]. User subroutines indicating the function to which the data

are to be fit are called by the main program which then iteratively attempts to determine the required variable coefficients (α and β in the log-normal case). The algorithm used selects an optimized correction vector for the coefficients by interpolating between the vector obtained by the gradient method and that obtained by a Taylor's series expansion truncated after the first derivative. Iteration is applied to this vector according to the least-squares method of estimating parameters until one of the several stopping criteria is met. Table 5 contains a list of parameters to which the curve-fitting was applied. Included in the table are the estimated sum of squares (ESS) and standard errors from the curve fitting results ($SE = \sqrt{ESS/118}$) and the coefficients for the respective log-normal equations.

Table 5. Curve Fitting Data

LOG-NORMAL CURVE FITS*				
	ESS	SE	α	β
CLASS A IO	1.06	.09	5.88	3.82
CLASS B IO	1.74	.12	3.04	2.64
CLASS C IO	.82	.08	9.62	3.19
CLASS A CPU	.25	.05	6.50	-9.60
CLASS B CPU	1.45	.11	2.47	1.32
CLASS C CPU	1.46	.11	8.88	1.50
LINES PRINTED	.55	.07	8.73	6.93

ESS: Estimated Sum of Squares
SE: Standard Error
α,β: log-normal curve coefficients
* log-normal curve fits were done on data transformed to the 0-1 interval

Simulated System Parameters

Having gathered the required frequency distributions and having determined the necessary percentage figures referred to earlier, these data were fed into the simulator and the simulator was run for a two hour period with the same job arrival rate as the benchmark run. Job statistics were collected over the benchmark run and a correlation was performed between the real and simulated data.

Correlation of Real and Simulated Runs

The parameters to be correlated fell into two separate categories. The first of these categories was the set of parameters from the real system that had actually been fed into the simulator in terms of frequency distributions. The parameters included in this category were I/O and CPU distributions for class A, B, and C jobs, and distributions of lines per job.

The second category of parameters to be correlated were those which were not fed directly into the simulator, but were produced by the simulator (and the real system) as a result of the characteristics of the input jobstream and the way in which the decisions about actually running that jobstream were made. The correlations in this second category particularly serve to validate the simulator, since their closeness verifies that indeed the simulator handles the jobstream in the same way the actual system does. The parameters in this category are frequency distributions of I/O, CPU and real time (time is O.S.) for the total jobstream, frequency

distributions for real time by class, a frequency distribution for units charged by job, and a frequency distribution of turnaround time for the total jobstream.

Correlation Coefficients for Frequency Distributions

In the correlation analysis [7], the various system parameters (number of I/O requests, number of kilobytes of core, etc.) were considered to be the independent variables in each case and the number of jobs using the respective resources were considered to be the dependent variables. The estimated coefficient of correlation, r, was used to test hypotheses about the population coefficient of correlation ρ. The coefficients may vary from -1 through 0 to +1. Perfect positive correlation (an increase in the components of one dependent variable vector determines exactly an increase in the components of the other dependent variable vector) yields a coefficient of +1. A perfect negative correlation (a decrease in the components of one vector determines a decrease in the components of the other) yields a coefficient of -1.

Tests of hypotheses about ρ, as well as about confidence intervals, can be made from moderately large samples if a function of r is used rather than the sample correlation coefficient itself. The function used is the Fisher r to Z transformation defined by

$$Z = 1/2 \ln \left(\frac{1 + r}{1 - r}\right) .$$

The sampling distribution of Z values is approximately normal, with an expectation given approximately by

$$E(Z) = \xi = 1/2 \ln \left(\frac{1 + \rho}{1 - \rho}\right) .$$

For moderately large samples, the hypothesis that ρ is equal to any value ρ_0 can be tested in terms of the test statistic:

$$\frac{Z - \rho}{\sqrt{1/(N - 3)}}$$

where N is the sample size and the test is referred to a normal distribution.

It is approximately true that for random samples of size N, an interval such as

$$Z - z_{(\alpha/2)} \sqrt{1/(N - 3)} \leq \xi \leq Z + z_{(\alpha/2)} \sqrt{1/(N - 3)}$$

will cover the true value of ξ with probability $1 - \alpha$. Here Z is the sample value corresponding to r, ξ is the Z value corresponding to ρ, $z_{(\alpha/2)}$ is the value cutting the upper $\alpha/2$ proportion in a normal distribution, and α is the conditional probability of rejecting the correct hypothesis in favor of an alternate hypothesis.

Since the coefficient of correlation is a measure of a linear relationship between sets of variables and since the Fisher r to Z transformation is designed for bivariate normal distributions, a transformation on the original data from both the real and simulated systems was necessary to make the parameter distributions approximately normal. The transformation applied was of the form

$$y' = \ln y$$

where y is the log-normal function of x as defined earlier. The α and β used in the y expression were those found using the curve fitting package.

An illustrative example of how this correlation technique would be applied to the class A I/O data distributions from the real and simulated systems will aid in clarifying the procedure.

A SOUPAC (Statistically Oriented Users Programming and Consulting) program package called "Correlation" is used to determine the estimated coefficient of correlation r [8]. Suppose that for class A I/O, r = .93. This high value of r evidences a very strong linear correlation between the two variables in question. Now, it is desired to test the hypothesis "does the number of class A jobs requesting 1-7, 8-14, . . ., 834-840 I/O requests in the real system and the linear relation account for more than 90 percent of the variance in the observed number of class A jobs making identical I/O requests in the simulated system?" That is, we actually want to thest the hypothesis

$$\rho^2 \geq .900 \text{ or } \rho \geq .948$$

against the alternative

$$\rho^2 < .100 \text{ or } \rho < .316$$

Here the Fisher r to Z transformation is used to find the test statistic

$$\frac{Z - E(Z)}{\sqrt{1/(N - 3)}} = \frac{1.658 - 1.811}{\sqrt{1/117}} = -1.65$$

In a normal sampling distribution, a standard score of 1.96 in absolute value is required for rejecting the hypothesis at the .05 level. Thus, we may safely conclude from this sample that more than 90 percent of the variance in the simulated system data is accounted for by its apparent linear relation to the real system data.

For $\alpha = .05$ and $z_{(\alpha/2)} = 1.96$, the 95 percent confidence interval for ξ is given approximately by

$$1.658 - 1.96(.0922) \leqq \xi \leqq 1.658 + 1.96(.0922)$$

$$1.478 \leqq \xi \leqq 1.838.$$

The corresponding interval for ρ is then approximately

$$.90 \leqq \rho \leqq .95.$$

We can assert that the probability is about .95 that sample intervals such as this cover the true value of ρ. Table 6 contains the list of the dependent variables which were correlated, along with the r, ρ and interval of confidence values for each.

Table 6. Correlations of Simulation Parameters

GROUP I. PARAMETER VALUES FED INTO SIMULATOR			
	r	ρ	Interval of Confidence (at .05 level)
Class A I/O	.22	.38	$.04 \leq \rho \leq .40$
Class B I/O	.42	.55	$.26 \leq \rho \leq .56$
Class C I/O	-.02	-	-
Class A CPU	.52	.64	$.38 \leq \rho \leq .64$
Class B CPU	.67	.76	$.55 \leq \rho \leq .76$
Class C CPU	-.02	-	-
Lines Printed	.72	.77	$.62 \leq \rho \leq .79$
GROUP II. PARAMETERS PRODUCED INDEPENDENTLY BY THE SIMULATOR			
Job I/O	.64	.73	$.52 \leq \rho \leq .73$
Job CPU	.69	.77	$.58 \leq \rho \leq .77$
Job Real Time	.41	.52	$.25 \leq \rho \leq .55$
Job Turnaround Time	.87	.90	$.82 \leq \rho \leq .90$
Job Cost	.67	.76	$.55 \leq \rho \leq .76$
Class A Real Time	.48	.58	$.33 \leq \rho \leq .61$
Class B Real Time	.37	.48	$.20 \leq \rho \leq .51$
Class C Real Time	0.0	-	-

Except for the class C coefficients, the parameters in Group I of Table 6 correlated quite satisfactorily. The class C jobs represented only 4 percent (16 jobs out of 399) of the total simulated jobstream and hence could not be expected to yield significant results. The class A I/O correlation was relatively low due to the "raggedness" of the frequency distribution of the real system class A I/O data at the lower end.

This unevenness could have been directly simulated, and hence the correlation made higher, by using the real frequency distribution data rather than the smoothed distribution. This tactic, however, which would yield more satisfactory results in this case, would contradict the effect of the original distribution smoothing technique which was felt to represent the whole population more accurately.

The Group II correlations in Table 6 were equally satisfactory. The real time and turnaround time coefficients in this group are values gathered from a second run of the simulator. The set of real time correlations from the first four-hour run were unacceptably low. The simulated system real times were generally much higher than the real system parameters and so the simulator was tuned by decreasing the number of milliseconds per I/O request and milliseconds of overhead time per job.

It should be noted that even though the turnaround correlation coefficient is high (.87), the means of the real and simulated distributions differ by about five minutes. This phenomenon occurs because the simulator does not simulate jobs that are queued for long periods of time waiting for special requests such as reversed-form printouts or India ink pen plots. Despite the fact that these jobs represent a small percentage of the total jobstream, their turnaround times can easily become one or two hours and thus influence the mean turnaround out of proportion to their number.

Correlation of Percentage Distributions

Those parameters which were evaluated in terms of percentage as opposed to frequency distribution included jobs having 1, 2, or 3 steps, I/O and CPU used by each step and jobs in classes A-C. Table 7 shows

the comparison of the percentage rates for these parameters for the simulated and real systems. This unusually high degree of agreement further emphasizes the close correspondence between the real and simulated systems.

Table 7. Percentage Distributions of Simulation Parameters

Percentage of:	Real System	Simulated System
Jobs with 1 step	43	40
" " 2 steps	54	51
" " 3 "	3	9
Total Job CPU used by step 1	38	39
" " " " " " 2	60	60
" " " " " " 3	2	1
Total Job I/O used by Step 1	41	41
" " " " " " 2	58	58
" " " " " " 3	1	1
Jobs in Class A	39	36
" " " B	56	60
" " " C	5	4

5.2. Analysis of Pay-for-Priority Simulated Run Results

Because of the satisfactory outcome of the simulator validating process, a reliable tool was available with which to test certain aspects of the specific pay-for-priority schemes. The eight models described earlier and summarized in Table 3 were each run for a four hour period

of simulated time. The simulation run results displayed in Table 9 and 10, and reasoned analysis in cases where simulation was not helpful, combined to form the comparative evaluation of the various schemes as presented in Table 8. In this table an attempt was made to classify a scheme's performance in meeting each objective as excellent, good, fair or poor.

Comments on the Significance of the Results

The GPSS simulation was used as a tool to collect various sets of data describing the performance of pay-for-priority models. The significance and limits of applicability of the results of the simulation runs should be understood before undertaking the task of studying the simulation results.

Frequency distributions of turnaround times were the major output of the runs. These distributions were collected over the nine categories of high, middle and low priority class A, B and C jobs. The data in Tables 9 and 10 represent the average of these distributions and as such are subject to the statistical phenomenon of one or two "stray" values distorting the average in a small sample size. Despite the fact that the overall sample size was greater than 700 jobs in every run, specific priority jobs within a class were likely to represent only a small percentage of the total jobstream. For example, there were only 5 high priority class C jobs that completed in the PRT runs. The average turnaround times listed in Table 10 for high priority class C jobs, therefore, are not significant. The seeming inconsistency in the fact that high priority

Table 8. Relative Merits of Pay-for-Priority Models

Objectives	Magic Number Schemes				PRT Schemes			
	Basic	3-level Priority	Ageing	Workload Factor	Basic	3-level Priority	Ageing	Workload Factor
1. User Control	P	G	E	E	P	G	G	G
2. System Efficiency	G	G	G	G	E	E	E	E
3. Load Leveling	F	G	G	E	F	G	G	E
4. Predictability	F	F	F	F	G	G	G	G
5. Easy Use	E	E	E	G	G	G	G	G
6. Practicality	E	E	E	E	F	F	F	F
7. Fairness	F	F	E	E	F	G	E	E

E - Excellent
G - Good
F - Fair
P - Poor

class C jobs spent a longer time in the system on the average than the middle priority class C jobs is thus not to be viewed as important in analyzing the effectiveness of the PRT priority assignment schemes.

In order to properly understand the significance of the turnaround time averages, the discussion of these averages in the earlier section on validation of the simulator should be recalled. It was noted then that even though the distributions of the real and simulated turnaround times correlated very closely, the means of these two sets of data differed. In fact, the simulated mean for turnaround time will always be lower than the actual turnaround time means. This feature of the simulator, however, in no way invalidates the results of the various runs, since their purpose was to study the relative performance of pay-for-priority models. Although absolute values of turnaround time are too optimistic in the simulated runs, the relative turnaround time values among the models are the important factor and may be compared with no loss of significance.

Understanding the method for generating jobstreams in the simulated runs will also aid in interpreting the simulator results. Even though the job arrival times and the resource assignments for jobs are taken from frequency distribution generators which are dependent on random numbers, the random numbers themselves were started from the same generating seed in all eight of the models. This implies that the identical jobstream presented itself for processing to each of the models. The difference in performance of the Magic Number and PRT schemes can, therefore, be attributed to the manner of choosing jobs from the jobstream and not to differences in the jobstreams themselves. The arrival of identical job-

streams to each of the models also explains why the turnaround time averages are the same for both the ageing and workload models in the Magic Number and PRT schemes. Since the workload factor feature does not affect job scheduling, but only charging for the job, jobs were run in exactly the same way in both cases, with the revenue collected being the distinguishing factor. Inconsistencies due to small sample size could also be expected to recur in each of the four runs of either of the two basic schemes.

In all of the simulation runs, the job arrival rate was chosen to be 200 jobs per hour. Thus, during the four hours of simulated time, approximately 800 jobs presented themselves to the system for processing. The job arrival rate of 200 jobs per hour was chosen to ensure a very busy system and thus be able to observe the various scheduling algorithms in an environment in which they should be able to perform most effectively. The figures for revenue collected were determined by using the charging algorithm presently in use by the Computing Services Office (see Appendix B).

An additional point to be made with regard to correctly interpreting the simulation results is that the turnaround time figures are valid for a particular set of parameters describing user priority requests. For these simulation runs, the assumption was that 30 percent of class A jobs would be high priority, 60 percent would be middle priority and 10 percent would be low priority. Similarly, 10 percent of class B jobs were assigned high priorities, 70 percent were assigned middle priorities and 20 percent were assigned low priorities. The values for high, middle and low class C jobs were 15 percent, 75 percent and 10 percent, respectively.

Finally, an explanation of the inconsistency apparent in the relatively long turnaround times of the high priority class A jobs under the PRT runs should be given. In the PRT schemes, a job was assigned a high priority by multiplying its normal PRT value by 5/6. This figure was chosen because it seemed to have an effect similar to that of moving a job to the next higher priority level as these levels were defined for the Magic Number scheme. Since class A jobs are generally very short, reducing high priority class A job PRTs by only 5/6 was not sufficient to place them at the front of the queue. The reduction will have to be adjusted to be greater for class A jobs to yield consistent results in the PRT schemes.

Discussion of the Magic Number Schemes

The Magic Number and PRT schemes present two different philosophies in running a jobstream. The Magic Number scheme is basically a discrete priority assignment which guarantees performance in terms of the class to which a job is assigned. Even though it is generally true that class A jobs run faster than class B jobs and class B jobs run faster than class C jobs, deviations from this running order are possible and, in fact, desirable in a priority oriented scheduling algorithm. In Table 9, it can be seen that high priority class B jobs run with a better average turnaround time than do low priority class A jobs. The same holds true for high priority class C jobs and low priority class B jobs. The philosophy in this case is that if a user is willing to pay for high priority, then his job should be run quickly, even at the expense of a higher class job.

Table 9 illustrates that the Magic Number schemes admirably perform this priority assignment function. The respective middle class turnaround times are approximately preserved across the table, regardless of the complexity of the scheme. This is important to the average user who should not be greatly affected by pay-for-priority features which he chooses not to use. But those jobs requesting a high priority receive excellent turnaround times for their respective class, while those requesting low priorities experience a longer than average turnaround time. In the ageing and workload factor schemes, for example, high priority class B jobs are able to run in about a tenth of the time it takes a middle priority class B job to run. High priority class A jobs experience half of the turnaround time of their middle priority counterparts, as do high priority class C jobs.

Another feature of the Magic Number schemes can be observed by comparing the simplest basic Magic Number scheme and the more complex schemes, proceeding from the addition of three levels of priority, to ageing, to the workload factor. Even the most complex pay-for-priority service can be offered with no decrease in resource utilization, revenue collected, or throughput.

Discussion of the PRT Schemes

Unlike the Magic Number schemes, the PRT schemes are continuous priority assignments, even though an association has been made between the PRT values and the standard class A, B, and C assignments for purposes

Table 9. Results from Magic Number Scheme Simulations

Average Turnaround in minutes	Magic Number Schemes			
	Basic	3-level Priority	Ageing	Workload Factor
High Class A	-	1.7	1.9	1.9
Middle Class A	4.4	3.8	4.0	4.0
Low Class A	-	7.6	6.4	6.4
High Class B	-	2.5	2.4	2.4
Middle Class B	22.6	6.1	23.3	23.3
Low Class B	-	84.9	30.0	30.0
High Class C	-	26.1	24.6	24.6
Middle Class C	46.0	56.2	52.4	52.4
Low Class C	-	-	-	-
CPU Utilization	.99	.99	.99	.99
Core Utilization	.73	.75	.73	.73
Revenue Collected	$1524.29	$1636.32	$1607.51	$1520.07
Jobs Processed	711	716	705	705

Table 10. Results from PRT Scheme Simulations

Average Turnaround in minutes	PRT Schemes			
	Basic	3-level Priority	Ageing	Workload Factor
High Class A	-	3.3	3.4	3.4
Middle Class A	2.2	1.7	1.4	1.4
Low Class A	-	2.9	2.7	2.7
High Class B	-	4.3	3.9	3.9
Middle Class B	5.7	5.4	5.1	5.1
Low Class B	-	7.0	6.8	6.8
High Class C	-	48.3	30.1	30.1
Middle Class C	29.2	25.9	25.6	25.6
Low Class C	-	50.5	50.5	50.5
CPU Utilization	.99	.99	.99	.99
Core Utilization	.75	.73	.74	.74
Revenue Collected	$1605.79	$1651.56	$1652.96	$1616.74
Jobs Processed	753	737	742	742

of comparison. Jobs are assigned a priority number based on their resource requests and are queued in HASP strictly according to increasing PRT. Any adjustments of jobs on the HASP queue, due to ageing for example, are done by changing a job's PRT and thus changing its position on the queue. The philosophy of priority assignment in the PRT scheme is to run the shortest jobs (smallest PRTs) first. Thus it can be seen in Table 10 that except for the previously mentioned discrepancies in the high priority class A and class C jobs, jobs are run with increasing PRT corresponding to constantly increasing turnaround times. In this case, in contrast to the Magic Number scheme, the equivalent of a low class A job is likely to run faster than the equivalent of a high class B job and a low class B job is likely ro run faster than the equivalent of a high class C job. Even the jockeying of jobs into artificial positions of low and high priorities appears not to degrade the basic performance of the PRT scheme, except in the case of the high priority class A jobs.

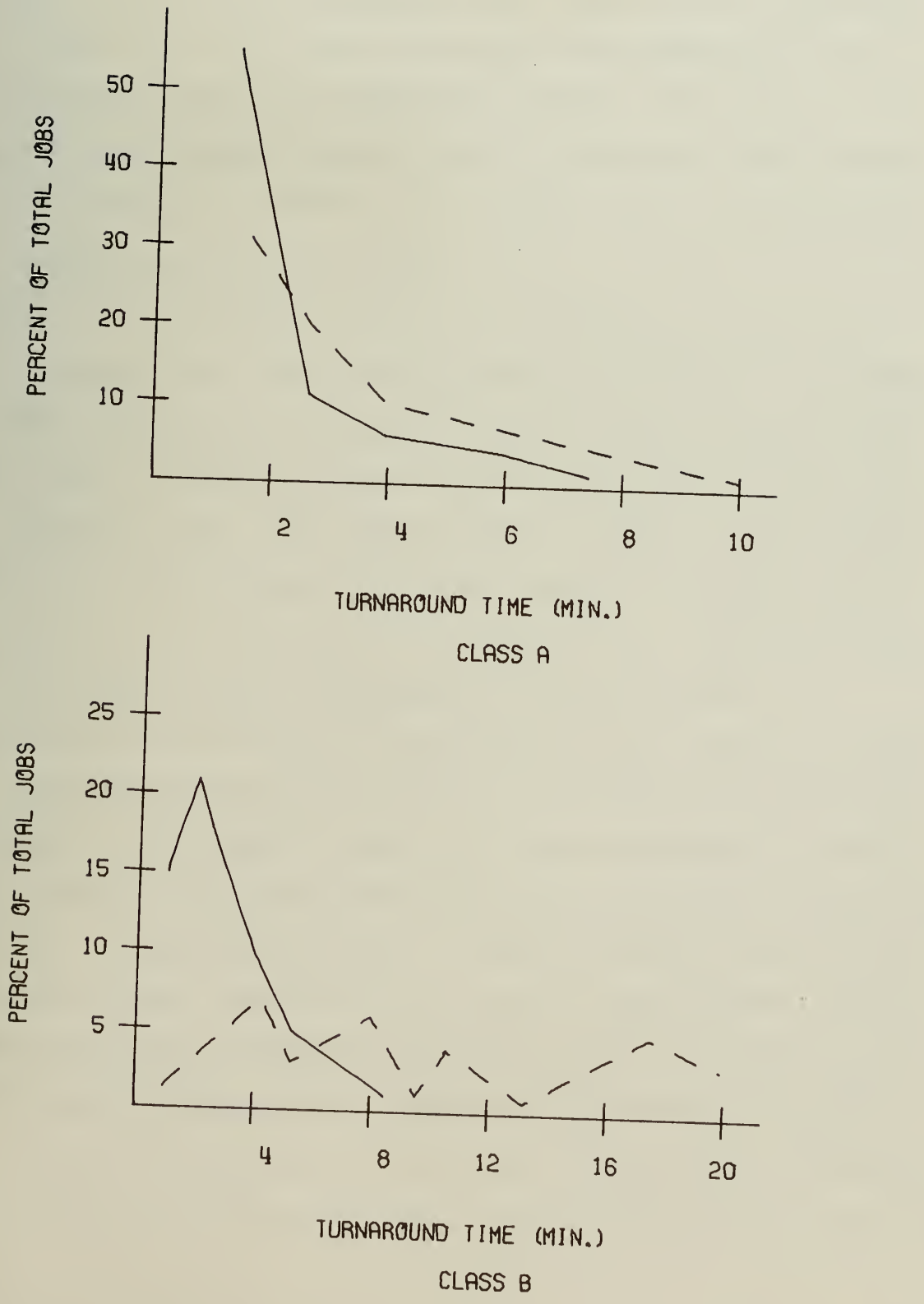
Because the philosophy of running the jobstream is different for the PRT schemes, the advantages to be seen from running the jobstream under these schemes are also different. The greatest asset of the PRT schemes is that they show a markedly improved turnaround time for all jobs that would lie in the class A or class B range. The frequency distributions for turnaround times in two corresponding Magic Number and PRT simulated runs are shown in Figure 3. The improved turnaround can easily be observed in both the class A and class B graphs, but it is most marked in class B where the average turnaround time for a class B job goes from 21 minutes under the Magic Number scheme to 5 minutes under the

PRT schemes. The total 399 class B jobs run in the Magic Number scheme represented 8440 minutes of turnaround time, while the 427 class B jobs processed in the PRT run required only 2288 minutes to run to completion. The class A statistics are impressive, with 283 jobs requiring 1041 minutes of turnaround time in the Magic Number scheme and 289 jobs requiring only 594 minutes of turnaround time in the PRT run.

In terms of HASP queue size, the PRT schemes out-performed the Magic Number schemes also. While the maximum class A, B, and C queue sizes were 85, 63, and 14 jobs, respectively, for the Magic Number run, the entire queue for the PRT scheme (including all of the waiting class A, B and C jobs) never exceeded 34 jobs. Thus at a job arrival rate of 200 jobs per hour, the PRT scheme kept the system significantly further from saturation.

A complete discussion of the effects of the PRT scheduling algorithm must include the observation that even though a smaller number of jobs remain on the HASP queue, this remaining set is likely to contain the jobs requesting the largest amounts of system resources. Since the PRT scheme queues jobs strictly according to size, the longest jobs will continue to be pushed to the rear of the queue as long as shorter jobs keep arriving. The ageing mechanism requires more than three hours to move such jobs forward in the queue and it is legitimate to ask at this point if this algorithm is effective merely because it chooses to run all the short jobs it can find and during the period over which the simulation was run it was not required to grind out some of the more demanding jobs.

Figure 3. Comparison of Magic Number and PRT Turnaround Times



Although this question cannot be answered conclusively without a longer simulation of the PRT scheme, some observations on data available from the present PRT runs indicate that this, in fact, was not the case.

First of all, the revenue collected in all of the PRT runs indicates that it consistently did more work than the Magic Number scheme. This observation is valid even while taking into account the fact that a part of the difference in the Magic Number and PRT revenue collected figures is due to an accumulation of more $1 cover charges in the PRT case. Secondly, a monitoring of the system workload during the PRT runs indicated that the system reached a peak period of busyness after about 2-1/2 hours of simulated time and then was able to recover from the overload and restore itself to a less busy system level. This observation indicates that the PRT scheme is indeed handling the jobs with very large resource requests.

A common criticism of the PRT scheme is that it relies very heavily on accurate user estimates of resources required for a job. The present Magic Number scheme does not reward users for making accurate estimates of required resources and, in fact, encourages them to request the largest values possible, given that their job will still remain in a certain class. When making accurate estimates is a behavior that is rewarded (jobs with smaller requests are placed higher on the HASP queue), then users will begin to make more accurate estimates. Precise estimates, in fact, add an additional, though subtle, priority level for the careful user.

As was the case with the Magic Number scheme, comparing the simplest basic PRT scheme and the most complex scheme with 3-level priority, ageing, and workload factor features leads to the conclusion that the most complex pay-for-priority service can be offered in this case also with no degradation in utilization, revenue or throughput.

Discussion of Predictor Results

Both the Magic Number and the PRT schemes had predictor mechanisms added to them in the manner described earlier. This predictor was intended to enable the user to obtain a projected turnaround time for his job, given its specific class (or PRT). Performance criteria were set up indicating acceptable limits of prediction error for jobs with different turnaround times. For example, it was decided that a good predictor should predict the turnaround time of a job that required less than 10 minutes in the system to within 3 minutes of that job's actual turnaround time. Similarly, the predictor should come to within 5 minutes in predicting turnaround time for a job that spends between 10 and 20 minutes in the system. Unfortunately, these requirements proved to be too rigid for the predictors in both the Magic Number and the PRT schemes. Table 11 shows the poor results of testing predictors against these measures of performance. The table is read in the following way: "for jobs spending X minutes in the system, the predictor was correct to within Y minutes Z percent of the time". For example, the first row reads "for jobs spending 10 minutes or less in the system, the predictor was correct to within 3 minutes 51 percent of the time for the Magic Number scheme and

68 percent of the time for the PRT scheme". Further, the results are not significant for tests over 40 minutes since less than 3 percent of the jobstream completed in 40 minutes or more. Less stringent predictor requirements were tested and are displayed in Table 12 in a format identifical to that of Table 11. Even though these results are better, they are still not satisfactory.

Table 11.

X	Y	Magic Number Z	PRT Z
10	3	51	68
20	5	29	16
30	7	0	0
40	10	0	0
60	12	0	0
>60	15	0	0

Table 12.

X	Y	Magic Number Z	PRT Z
10	5	64	82
20	10	43	46
30	15	0	0
40	20	100	33
60	30	0	20
>60	30	15	0

It should be noted that even though these particular prediction schemes were unsuccessful, it is not to be implied that all schemes will be unsuccessful. One of the limiting factors in making predictions is the lack of recent information on jobs that run less frequently than others. For example, high priority class A jobs completed at a rate of one every three minutes. Thus it is feasible that turnaround time predictions for high priority class A jobs could be about 5 minutes old, and the system

history reflected in this figure could be no longer accurate. An alternative approach to prediction is to prepare tables of average and worst-case turnaround figures for each of the classes under varying workloads. These tables could be referenced by the user and would offer a more stable reference point. More experimentation is required in this area if adequate predictability of a pay-for-priority scheme is to be realized.

6. CONCLUSIONS

Recommendations

A consideration of the work involved in the implementation of any variations of the Magic Number or PRT schemes is essential before a decision can be made on which of the schemes to adopt. All of the Magic Number schemes are inherently easier to implement because the basic Magic Number mechanisms of class assignments and priority within a class assignment are already present in the current system. The addition of the three levels of priority--high, middle and low--would simply involve another parameter on the ID card and the logic to assign a job a priority 5, 6, or 7, depending on whether the job had requested low, middle or high priority within its class. The means for charging some percentage of actual job cost are present in the system charging mechanism and would have to be activated. The ageing mechanism is also already present in HASP and only constant parameters (exactly how long to let jobs age within a given priority level) would have to be managed. Addition of the workload factor would require storing average turnaround times of various classes and subsequently using these figures as the basis for deciding system activity and the price of priority. The logic to do this would have to be added to the present system.

The basic system change required to implement the PRT scheme would be the addition of a new queue add and queue delete module to HASP. Because the PRT assignments are continuous, it is necessary to queue jobs in the order of increasing PRT. Logic would also have to be added which would calculate the job's PRT given its core, I/O and CPU requests.

Priority levels and ageing modifications are accomplished by changing a job's PRT and thus changing its position on the queue. Given the queue add and queue delete module, these changes would be straightforward pieces of logic. The addition of the workload factor would require considerably more work than the other modifications since the workload factor is determined from the slope of the predictor curve and the predictor curve itself was found to perform unacceptably. If it were decided to implement the PRT with workload factor, experimentation with a more accurate predictor should be carried on simultaneously with the initial work for the PRT implementation so that the improved predictor could be ready for use when it was required.

The impressive improvement in turnaround time and throughput for the PRT scheme is a factor far outweighing the more complex coding required to implement this scheme. Yet, undertaking the complex coding may impose an undue delay on the introduction of a pay-for-priority scheme. The plan for implementation, therefore, should consist of several phases. The first of these phases is an introduction of the three levels of priority

into the present Magic Number scheme, ignoring ageing and the workload factor. Following this, the Magic Number formula should be changed to the PRT formula, but class and initiator settings should remain essentially the same. Subsequent phases should include the changeover to a continuous PRT scheme, the addition of the ageing module and the addition of the workload factor module. Beside providing for quick implementation with a potential for progressively more complex improvements, this plan also nicely provides for the monitoring of system and user response after each individual system change is made.

The ultimate decision for the implementation of a specific pay-for-priority algorithm rests with administrative personnel. The type of information needed to make this decision includes answers to such questions as how will this scheme effect system efficiency, revenue collected, and the user community. Some of the answers to these questions can only be determined after implementation of a scheme. Many important questions, however, can be answered prior to implementation by using system models to measure system performance.

The results of the study of pay-for-priority schemes as presented in this paper verify the usefulness of simulation as an adjustable model which is able to easily incorporate features of various scheduling algorithms. Further, the performance measurements obtainable from simulation, combined with reasoned analysis by experienced systems analysts can provide accurate and adequate information for input into the decision making process which precedes system changes. Thus, a responsible introduction of pay-for-priority scheduling is possible.

The Road Ahead

The development, evaluation and recommendation of a pay-for-priority scheme to be implemented on the IBM 360/75 is only the first step in a series of events which will lead to the successful adaptation of such a scheme. After actual implementation is accomplished, ongoing evaluation and maintenance of the scheme are natural and essential subsequent tasks, particularly in the scheme's first few months of use.

Implementation of a pay-for-priority scheme requires two important tasks. The first of these is the laborious job of coding the scheme that has been selected into the HASP and O.S. software configuration. A second and equally important part of implementation involves user orientation. The computing community should thoroughly understand the scheme and feel confident that they can manipulate it to their own advantage.

Dynamic evaluation and maintenance of the scheme should play a major role in the first few months of use of the pay-for-priority scheme. The values of various parameters in all the recommended schemes were chosen on the basis of being "reasonable predictions" of user and system behavior. These parameters include charges for various priority levels, percentage of users requesting high or low priorities in each class, ageing times and the workload factor h. User response must be closely monitored, therefore, and answers must be sought for such questions as:

--what percentage of users choose high or low priorities and is this an acceptable level of use?

--is the amount being charged for high and low priorities serving as an incentive for their respective use?

--what are acceptable waiting times to be incorporated into the ageing section of the algorithm?

--is the predictor calculating turnaround times that correlate closely with the actual turnaround values?

--is the workload factor h an acceptable one?

--is the revenue being collected sufficient to justify the scheme?

--what user suggestions could make the scheme more acceptable to the whole user community?

The answer to these questions should be used to more finely tune the scheme that is adopted, using whatever means are necessary to successfully do the tuning. These "means" may range from a straightforward change of a particular constant parameter to reruns of the pay-for-priority simulator to test more feasible alternatives.

The GPSS simulator has played an important role in the recommendation of a pay-for-priority scheme. The full circle of validation for the simulator must be completed as an extension of the pay-for-priority project. The circle began with the development of a benchmark jobstream from which to gather data to feed into the simulator. The simulator was tuned to fit the real system data and then it was used to make predictions about a typical jobstream. As a final step, after the pay-for-priority scheme has been implemented, the benchmark should be run under the scheme and the jobstream performance should be rigorously compared to the simulator predictions for it. This analysis will be the ultimate test for the validity of the simulator.

The results of the simulated runs of the various pay-for-priority schemes have raised questions and problems for further study that were not anticipated. These results suggest that more thought and effort will be required to develop predictors that perform satisfactorily within acceptable limits. If it is decided to adopt the PRT scheme, more extensive simulated runs of that scheme should be performed to guarantee that its striking advantages are maintained over longer periods of time. Further experimentation should be done with varying the parameters that indicate what percentage of the user community will actually request high or low priority runs. At what point will a particular job mix begin to cause system efficiency degradation? Other areas of observation present themselves, such as monitoring the variations in queue sizes as the simulation progresses. If the Magic Number scheme is to be used, its components and relevance should be critically examined. A glaring omission is its ignorance of maximum core requests, a factor appropriate to consider for scheduling and included in the PRT calculations.

Finally, another natural and desirable extension of the pay-for-priority project is the further development of a benchmark jobstream. Four hours of sampling was done to formulate a jobstream for use in developing the pay-for-priority scheme. Another four-hour sample should be taken and the two jobstreams should be compared, combined and used for a finer characterization of jobs as they are serviced by the Computing Services Office.

APPENDIX A: IBM 360/75 SCHEDULING ALGORITHM

Under the present HASP and O.S. system, jobs are read simultaneously from terminals, tapes, readers, disks, and other devices. Upon entering the system, jobs are assigned a "Magic Number", Y, where the value of Y is determined as follows: Y = SEC + .1*IOREQ + .03*LINES . SEC represents seconds of CPU usage, LINES represents printed output, and IOREQ represents the transfer to or from core storage of blocks of data. Based on this Magic Number, a "class" assignment is given to each job. Class boundaries have traditionally been set at approximately $Y \leq 250$ for class A, $250 < Y \leq 750$ for class B, $750 < Y \leq 3000$ for class C and $Y > 3000$ for class D. A special class of very small fast jobs is called "express" or class X jobs.

Any one of seven initiators can be set to recognize up to five different classes of jobs, in a specific order. It is in this order that a free initiator will take a job off the spool and feed it to O.S. For example, if an initiator is set CBA, it will first search the spool for a class C job; if not found, it will look for a class B. If there is no B job, and no A job either, the initiator will be put in a wait state. Once the job is selected, it is put on the O.S. queue to be serviced by the operating system. After a job is placed on the O.S. queue, there is no longer any class distinction. Another set of initiators selects jobs on a first-come, first-served basis and removes them from the O.S. queue.

It is the function of these initiators to take the job through the various stages of execution. Jobs are selectively given control of the CPU by the O.S. Scheduler. The job with the highest dispatching priority (the job that has the lowest ratio of CPU time to I/O requests will get the highest dispatching priority) is given control until an interrupt occurs - either user initiated or system initiated.

APPENDIX B: JOB COSTS

Central Processor Charges

360/75 (calculated for each job step):

units charged = 0.04(X+Y) (0.0045Z+0.5)

X = CPU time in centiseconds (0.01 sec.)
Y = number of Input and Output requests
Z = core size used in Kilobytes

The CPU charge for the entire job is $.01/unit charged.

Overhead Charge

HASP	$ 1.00 per job submitted
EXPRESS	1.00 per job submitted

On-Line Peripheral Charges

Disk drive (permanent	$0.0065	per track per day
Disk/Tape drive (setup)	1.00	per volume mounted
Card Reader	1.40	per thousand cards read
Card Punch	3.90	per thousand cards punched
Line Printer	.80	per thousand lines printed
CalComp Plotter	20.00	per hour plotter time
2741/TTY terminal	1.00	per hour connect time

LIST OF REFERENCES

[1] Simpson, T. H., "Houston Automatic Spooling Priority System - II (version 2)," International Business Machines Corporation, 1969.

[2] Salz, F., "A GPSS Simulation of the 360/75 Under HASP and O.S. 360," Department of Computer Science Report No. UIUCDCS-R-72-528, University of Illinois at Urbana-Champaign, June 1972.

[3] _____, Introduction to OS/VS2 Release 2, GC28-0661-0, International Business Machines Corporation, 1973.

[4] Mamrak, S., "Proposed Priority Schemes for the IBM 360/75," Noncirculated internal document, Department of Computer Science, University of Illinois at Urbana-Champaign, February 1973.

[5] Salz, F., "Simulation Analysis of a Network Computer," Master of Science Thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, June 1973.

[6] Middleton, J. A., "Least Squares Estimation of Non-Linear Parameters -NLIN," 360D-13.2.003, International Business Machines Corporation, 1968.

[7] Hays, W. L., Statistics, Holt, Rinehart and Winston, Inc., New York, 1963, pp. 529-536.

[8] _____, "Correlation," SOUPAC Program Descriptions, Report No. 370-3, University of Illinois at Urbana-Champaign, March 1, 1971.

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUCDCS-R-73-605	2.	3. Recipient's Accession No.

4. Title and Subtitle

Simulation Analysis of a Pay-For-Priority Scheme for the IBM 360/75

5. Report Date: August 1973

6.

7. Author(s): Sandra Ann Mamrak

8. Performing Organization Rept. No. UIUCDCS-R-73-605

9. Performing Organization Name and Address
University of Illinois at Urbana-Champaign
Department of Computer Science
Urbana, Illinois 61801

10. Project/Task/Work Unit No.

11. Contract/Grant No.: NSF GJ 28289

12. Sponsoring Organization Name and Address
National Science Foundation
1800 G Street, N.W.
Washington, D.C. 20550

13. Type of Report & Period Covered: Master of Science Thesis

14.

15. Supplementary Notes

16. Abstracts When a computing facility becomes so heavily utilized that dissatisfaction about long turnaround times is prevalent in the user community, the introduction of a priority consideration into the scheduling algorithm will aid in lessening the problem. Users who desire shorter turnaround time may then opt for a high priority position on the job queue and users with less urgent jobs may opt for a middle or a low priority position on the job queue. High and low priority queue positions may be associated with higher and lower rates of job cost respectively. The results of the development and evaluation of a priority scheduling algorithm for the University of Illinois computing center are presented in this paper. Project goals are set forth, algorithm options are considered and a thorough discussion of the formulation and analysis of various simulation models of proposed schemes is included. Recommendations are made concerning the follow-up procedures required to make the project successful.

17. Key Words and Document Analysis. 17a. Descriptors

Priority Scheduling; Simulation; Validation

b. Identifiers/Open-Ended Terms

c. COSATI Field/Group

18. Availability Statement: Release Unlimited

19. Security Class (This Report) UNCLASSIFIED

20. Security Class (This Page) UNCLASSIFIED

21. No. of Pages: 60

22. Price: ------